Machine Learning tutorial

Takaya KOIZUMI

Mathematical Science, B4

Applied Mathematics and Physics informal seminor

Contents

Contents

機械学習とは

機械学習とは, 「関数近似論」である.

世の中で機械学習を使って実現したと言われている技術

1. 翻訳 ({ 全ての日本語 }→{ 全ての英語 } という関数)
2. メール分類 ({ 全てのメールの文章 }→{ 迷惑メール, 非迷惑メール } という関数)
3. 音声認識 ({ 音声 }→{ 文章 } という関数)

もちろん, 間違いを起こすこともある. (大事なメールが, 迷惑メールに入ることも...)

数学的には

前スライドの話を集合論を用いて, もう少し数学的にきちんと書くならば, 以下のようになるだろう.

機械学習?

$\mathcal{X}, \mathcal{Y}$ をそれぞれ $\mathbb{R}^d$, $\mathbb{R}^m$ の部分集合とする. この時, 良い関数 $f : \mathcal{X} \to \mathcal{Y}$ を見つけることを機械学習という.

しかし, この定義には以下の問題がある.

上の定義の問題点

1. 候補となる関数が多すぎる. (ヒントも何もないのに探せない)
2. 良い関数とは何か, 定義されていない.

Contents

前半の問題解消

では, まず前半の「候補となる関数が多すぎる.」という問題を解決していこう.
この問題の解決方法として, 人間がヒント (条件) を与えてあげることで, 関数全ての集合ではなく, ある程度絞った集合 $\mathcal{H}$ にするということを考える. この $\mathcal{H}$ のことを仮設空間 (Hyposesis space) と呼ぶ.

Definition (仮設空間)

$\mathcal{X}, \mathcal{Y}$ をそれぞれ $\mathbb{R}^d, \mathbb{R}^m$ の部分集合とする. この時, 集合

$$\mathcal{H} := \{f_w : \mathcal{X} \to \mathcal{Y} \mid f_w \text{ に関する条件 }\}$$

のことを仮設空間と呼び, $\mathcal{X}$ を特徴量空間, $\mathcal{Y}$ をラベル空間と呼ぶ. また, $f_w \in \mathcal{H}$ を仮設と呼ぶ.

後半の問題解消

では, 後半の「良い関数」というものを定義していこう. 機械学習において, 良い関数とは, 未知のデータ X に対して正しい値 Y を返す関数である. そのために, 関数 f に対してその良さを表す指標である汎化誤差を定義する.

Definition (汎化誤差, 損失関数)

$\mathcal{H}$ を仮設空間, $(\Omega, \mathcal{F}, \mathbb{P})$ を確率空間, ρ をデータの確率分布とする. この時, 汎化誤差 $\ell : \mathcal{H} \to \mathbb{R}$ を,

$$\ell(f_\theta) = \mathbb{E}_{(X,Y)\sim\rho}[l(f_w(X), Y)]$$

と定義する. ここで, $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ は損失関数と呼ばれる凸関数である.

損失関数の具体例

損失関数

ここで, よく使われる損失関数の例をいくつか述べておく.

1. 2 乗損失関数 $l(y_1, y_2) = (y_1 - y_2)^2$
2. 交差エントロピー誤差 $l(y_1, y_2) = -y_2 \log y_1$

これで, 「良い関数」を作るためには, 汎化誤差 ℓ を最小化させるような仮設空間 $\mathcal{H}$ の元 f を見つければ良いと言うことになったわけだが, 汎化誤差には期待値が含まれるため, 直接最適化させることが難しい. そのため, 持っているデータを利用して別の関数を用意し, その関数を最小化することを考える.

データと経験損失関数

Definition (データ)

$(\Omega, \mathcal{F}, \mathbb{P})$ を確率空間, ρ をデータの確率分布とする.
$\{(X_n, Y_n)\}_{n=1}^N$ を ρ に従う独立な確率変数列とした時,
$\{(X_n, Y_n)\}_{n=1}^N$ の観測値 $\{(X_n(\omega), Y_n(\omega))\}_{n=1}^N$ のことをデータ *(Data)* と呼び, $D = \{(x_n, y_n)\}_{n=1}^N$ と表記する.

Definition (経験損失関数)

$\mathcal{H}$ を仮設空間, $D = \{(x_n, y_n)\}_{n=1}^N$ をデータ, $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ を損失関数とする. この時, 経験損失関数 $\mathcal{L}_D : \mathcal{H} \to \mathbb{R}$ を,

$$\mathcal{L}_D(f_w) = \sum_{n=1}^{N} l(f_w(x_n), y_n)$$

と定義する.

機械学習と学習アルゴリズム

さて, ここで改めて機械学習を定義しよう.

Definition (機械学習, 学習アルゴリズム)

$\mathcal{H}$ を仮設空間, D をデータ, $\mathcal{L}_D$ を経験損失関数とする. この時, アルゴリズム $\mathcal{A}$ を用いて, $\mathcal{L}_D$ を最小化・最大化させる過程のことを機械学習 (あるいは単に学習) と呼び, その時のアルゴリズム $\mathcal{A}$ のことを学習アルゴリズムと呼ぶ. また, 最適解 $f^* \in \mathcal{H}$ を最適仮設と呼び, その時のパラメータ w^* を最適パラメータと呼ぶ.

これ以降, 5 つ組 $(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{L}_D)$ を ML 空間と呼ぶことにする.

Contents

最も基礎的なモデル

まず, 最も基礎的な機械学習モデルである単回帰分析を紹介する.

Example (単回帰分析)

ML 空間 $(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{L}_D)$ を以下で定義する. $\mathcal{X} = \mathbb{R}$, $\mathcal{Y} = \mathbb{R}$,

$$\mathcal{H} = \{f : \mathcal{X} \to \mathcal{Y} \mid f(x) = wx, w \in \mathbb{R}\},$$

$$\mathcal{L}_D(f) = \sum_{i=1}^{N} (f(x_i) - y_i)^2.$$

この ML 空間 $(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{L}_D)$ 上で,

$$\arg\min_{f \in \mathcal{H}} \mathcal{L}_D(f)$$

を求める問題を単回帰分析という.

SRA の学習

SRA は勾配降下法などを用いて解くこともできるが, 今回は解析的に最適パラメータを求める. SRA の経験損失関数は w に関して 2 次関数となっているので, 平方完成を用いると,

$$f^*(x) = w^* x, \qquad w^* = \frac{\sum_{i=1}^{N} x_i y_i}{\sum_{i=1}^{N} x_i^2}$$

と定義される $f^* \in \mathcal{H}$ が最適であることがわかる.

Contents

重回帰分析

Example (重回帰分析)

ML 空間 $(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{L}_D)$ を以下のように定義する.
$\mathcal{X} = \mathbb{R}^d (N \geq d)$, $\mathcal{Y} = \mathbb{R}$,

$$\mathcal{H} = \{f : \mathcal{X} \to \mathcal{Y} \mid f(x) = W^\top x + b, W \in \mathbb{R}^d, b \in \mathbb{R}\},$$

$$\mathcal{L}_D(f) = \sum_{i=1}^{N} (f(x_i) - y_i)^2.$$

この ML 空間 $(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{L}_D)$ 上で

$$\arg\min_{f \in \mathcal{H}} \mathcal{L}_D(f)$$

を求める問題を重回帰分析という.

MRA の学習

SRA の時と同様に解析的に最適パラメータを求める (簡単のために $b = 0$ とする). $X \in \mathbb{R}^{N \times d}$, $\mathrm{y} \in \mathbb{R}^N$ を $X = [x_1 x_2 \dots x_N]^T$, $\mathrm{y} = (y_1, y_2, \cdots, y_N)^\top$ と定義する. もし X が可逆であるとすると,

$$f^*(x) = W_*^\top x, \qquad W_* = (X^\top X)^{-1} X^\top \mathrm{y}$$

と定義される $f^* \in \mathcal{H}$ が最適であることがわかる. なお, $b \neq 0$ の場合もデザイン行列 X を変更し, $w_0 = b$ とすることで, 上記の計算の場合に帰着することができる [2].

Contents

基底関数

この節では, はじめに多項式回帰を紹介する. その前に基底関数というものを導入する.

Definition (基底関数)

$\mathcal{X}$ を $\mathbb{R}$ の部分集合とする. $\mathcal{X}$ から $\mathcal{X}$ への C^1 級関数列 $\{\phi_n\}_{n=1}^{d}$ が $\mathbb{R}$ 上 1 次独立である時, $\Phi(x) = (\phi_0(x), \phi_1(x), \ldots, \phi_d(x))$ で定義される $\Phi : \mathcal{X} \to \mathbb{R}^{d+1}$ を基底関数と呼ぶ.

基底関数を用いることで, 非線形なデータにも対応することができる.

多項式回帰

Example (多項式回帰)

ML 空間 $(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{L}_D)$ を以下のように定義する.
$\mathcal{X} = \mathbb{R}, \mathcal{Y} = \mathbb{R},$

$$\mathcal{H} = \{f : \mathcal{X} \to \mathcal{Y} \mid f(x) = W^\top \Phi(x), W \in \mathbb{R}^{d+1}\},$$

$$\mathcal{L}_D(f) = \sum_{i=1}^{N} (f(x_i) - y_i)^2.$$

ここで, $\phi_n(x) = x^n$, $n \in \{0, 1, \cdots, d\}$ とする.
この ML 空間 $(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{L}_D)$ 上で

$$\arg\min_{f \in \mathcal{H}} \mathcal{L}_D(f)$$

を求める問題を多項式回帰という.

多項式回帰での学習

今回も解析的に解くことにする.
$X = [\Phi(x_1), \Phi(x_2), \cdots, \Phi(x_N)]^T \in \mathbb{R}^{N\times d}$ とし,
$\mathrm{y} = (y_1, y_2, \cdots, y_N)^T \in \mathbb{R}^N$ とする. この時, 最適仮説 f^* は重回帰分析と同様に $W = (X^\top X)^{-1} X^\top \mathrm{y}$ とすれば

$$f^*(x) = W^\top x$$

である.

学習パラメータとハイパーパラメータ

多項式回帰の多項式の次数 $d \in \mathbb{N}$ や基底関数 $\{\phi_n\}_{n=1}^d$ のようにコンピュータに学習させるのではなく, 機械学習モデルの設計者が設定するパラメータのことをハイパーパラメータと呼ぶ. 一方, $W \in \mathbb{R}^{d+1}$ のように, データからコンピュータが自動で学習するパラメータのことを学習パラメータと呼び, Θ と表す.

多項式回帰の過学習

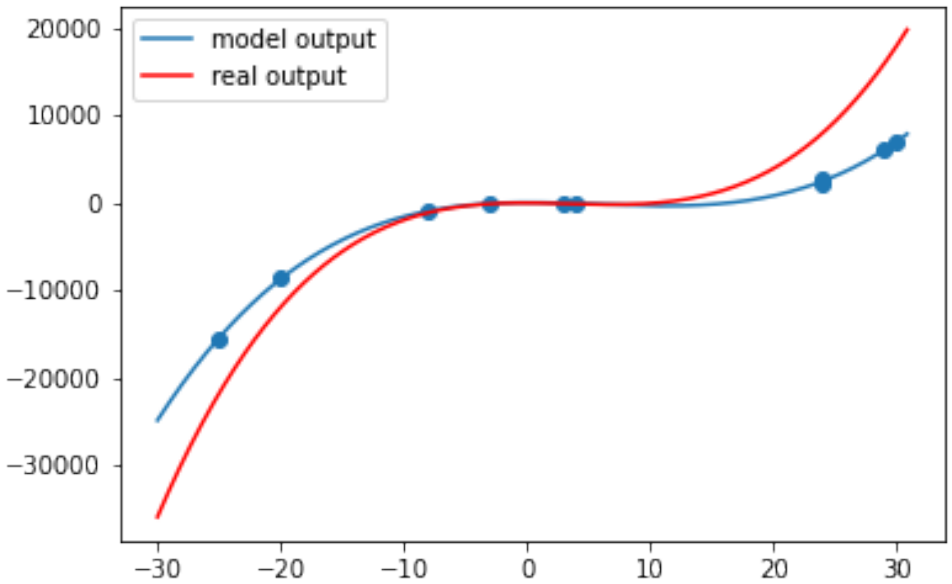


Figure: $d = 3$ の時の多項式回帰

多項式回帰で学習を行うと左の図のように，「教師データには適合しているが，未知のデータには全く対応できない」モデルが学習されてしまう．
このように，訓練誤差に対して，汎化誤差が大きくなってしまうことを**過学習**と呼ぶ．

Contents

正則化

過学習を防ぐために, 学習パラメータを制限する方法のことを正則化 (regularization) と呼ぶ. 具体的には経験損失関数に正則化項というものを加えて, パラメータが大きくなり過ぎないようにする.

Definition (正則化)

$\mathcal{L}_D : \mathcal{H} \to \mathbb{R}$ を経験損失関数とする. $\mathcal{L}_D^R := \mathcal{L}_D + L^R$ とする. この時, $\mathcal{L}_D^R$ を $\mathcal{L}_D$ の正則化と呼び, $L^R : \Theta \to \mathbb{R}$ を正則化項と呼ぶ.

Ridge 正則化多項式回帰

Example (Ridge 正則化多項式回帰)

$\lambda \in \mathbb{R}^+ := \{x \in \mathbb{R} \mid x \geq 0\}$, $d \in \mathbb{N}$ を任意にとる. 多項式回帰の *ML* 空間 $(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{L}_D)$ の経験損失関数を

$$\mathcal{L}_D(f) = \sum_{i=1}^{N} (f(x_i) - y_i)^2 + \lambda W^\top W$$

と正則化する. *(*ここで, $\lambda W^\top W$ が正則化項である*.)* この *ML* 空間 $(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{L}_D)$ 上で

$$\arg\min_{f \in \mathcal{H}} \mathcal{L}_D(f)$$

を求める問題を *Ridge* 正則化多項式回帰という.

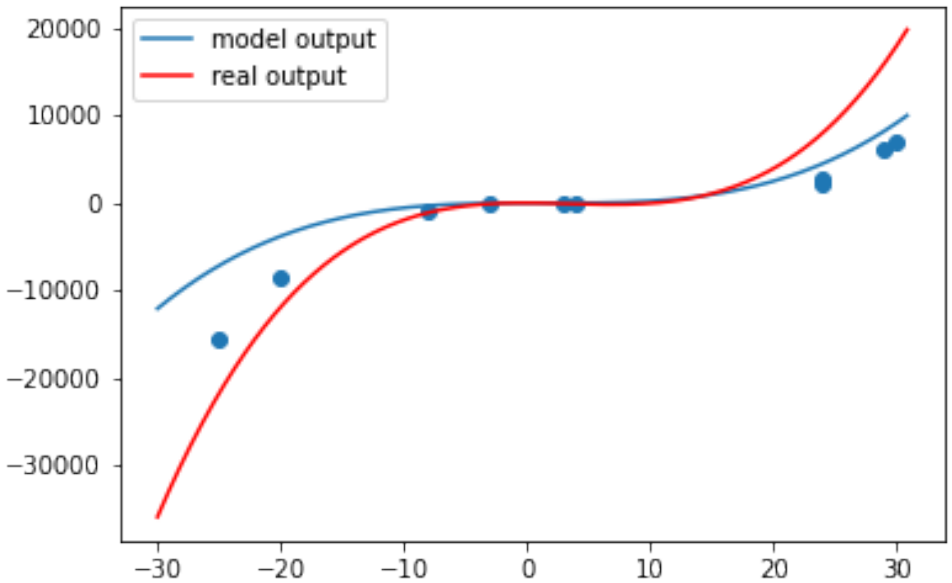
Ridge 多項式回帰での学習

$X = [\Phi(x_1), \Phi(x_2), \cdots, \Phi(x_N)]^T \in \mathbb{R}^{N \times d}$ とし,
$\mathrm{y} = (y_1, y_2, \cdots, y_N)^T \in \mathbb{R}^N$ とする. この時, 最適仮説 f^* は

$$f^*(x) = W_*^\top x, \quad W_* = (X^\top X + \lambda I)^{-1} X^\top \mathrm{y}$$

となる. (ここで, I は単位行列である.)

正則化の実験



正則化を行うことで, データに完全に fit せず, 未知のデータにもある程度対応できるようになった. ただ, -10 以下のデータに関しては逆に精度が下がる結果になってしまった. (方程式の方が綺麗...)

Figure: $d = 3$ の時の Ridge 正則化多項式回帰

Contents

分類問題と one-hot ベクトル

今回は入力データがどのクラスに所属するのかを予測するモデルである「分類」モデルを扱う. そのために, 準備としてソフトマックス関数と one-hot ベクトルを導入する.

Definition (ソフトマックス関数)

$\psi : \mathbb{R}^d \to \mathbb{R}^d$ を以下で定義する.

$$\psi(x)_i = \frac{\exp(x_i)}{\sum_{k=1}^{d} \exp(x_k)}$$

この時, ψ をソフトマックス関数と呼ぶ.

Definition (one-hot ベクトル)

$y \in \mathbb{R}^m$ が第 $c \in \{1, 2, \cdots, m\}$ 成分が 1 であり, 残りの成分が 0 である時, y を *one-hot* ベクトルという.

クラスラベルと one-hot ベクトル

one-hot ベクトルの使い方について解説する.
ジャンケンの手の画像が与えられた時にその手が「グー, チョキ, パー」のうちどれなのかを予測するモデルを構築するとする $(m = 3)$. この時, データセット $\{(x_n, y_n)\}_{n=1}^N \subset \mathbb{R}^{28\times 28} \times \{\text{グー}, \text{チョキ}, \text{パー}\}$ を

1. $y_i =$ "グー" なら $y_i = (1, 0, 0)$
2. $y_i =$ "チョキ" なら $y_i = (0, 1, 0)$
3. $y_i =$ "パー" なら $y_i = (0, 0, 1)$

として変更し, 新たなデータセット $\{(x_n, y_n)\}_{n=1}^N \subset \mathbb{R}^{28\times 28} \times \mathbb{R}^3$ を得る.

Contents

ロジスティック回帰

Example (m 値分類ロジスティック回帰)

ML 空間 $(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{L}_D)$ を以下のように定義する.
$\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \mathbb{R}^m$,

$$\mathcal{H} = \{f : \mathcal{X} \to \mathcal{Y} \mid f(x) = \psi(Wx + b), W \in \mathbb{R}^{m \times d}, b \in \mathbb{R}^m\},$$

$$\mathcal{L}_D(f) = -\sum_{n=1}^{N}\sum_{k=1}^{m} y_{nk} \log f(x_n)_k,$$

ここで $\psi : \mathbb{R}^m \to \mathbb{R}^m$ はソフトマックス関数であり, y_n は *one-hot* ベクトルである. この ML 空間 $(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{L}_D)$ 上で

$$\arg\min_{f \in \mathcal{H}} \mathcal{L}_D(f)$$

を求める問題を m 値分類ロジスティック回帰という.

勾配降下法

今回は解析的に, 経験損失関数の解を求めるのではなく, 勾配降下法というアルゴリズムを用いて, 解くことにする.

Algorithm 1 Gradient Decent

Require: F: C^1 function on $\mathbb{R}^d$
Require: $0 < \alpha < 1$: learning rate
Require: θ: Initial parameter vector
  $\theta \leftarrow \theta_0$
  while θ not converged **do**
    $\theta \leftarrow \theta - \alpha \nabla F(\theta)$
  end while
  return θ

経験損失関数の勾配計算

今回の経験損失関数に対して w_{ij}, b_i についての偏微分を計算すると以下のようになる.

$$\frac{\partial \mathcal{L}_D}{\partial w_{ij}} = -\sum_{n=1}^{N} (y_{ni} - f(x_n)_i)\, x_{nj}, \quad \frac{\partial \mathcal{L}_D}{\partial b_i} = -\sum_{n=1}^{N} (y_{ni} - f(x_n)_i)$$

したがって, $X = [x_1, x_2, \cdots, x_N]$, $Y = [y_1, y_2, \cdots, y_N]$, $F = [f(x_1), f(x_2), \cdots, f(x_N)]$, $1 = [1, 1, \cdots, 1]^\top \in \mathbb{R}^{N\times 1}$ とすれば,

$$\frac{\partial \mathcal{L}_D}{\partial W} = -(Y - F)X^\top, \quad \frac{\partial \mathcal{L}_D}{\partial b} = -(Y - F)1$$

となる.

クラスラベルの予測方法

ロジスティック回帰の最適仮説 f^* が得られた時, $x \in \mathcal{X}$ に対してのクラスラベル $c \in \{1, 2, \cdots, m\}$ を,

$$c = \underset{i \in \{1,2,\cdots,m\}}{\arg\max} \ f(x)_i$$

として求める. これは, シグモイド関数の出力 $\psi(x)$ の i 番目の成分 $\psi(x)_i$ が x がクラス i に所属する確率であるとみなせることから, 所属する確率が一番高いものを予測値とするのが自然だからである.

Contents

MNIST での学習

MNIST とは以下のような 28×28 ピクセルの数字か書かれたデータセットである.

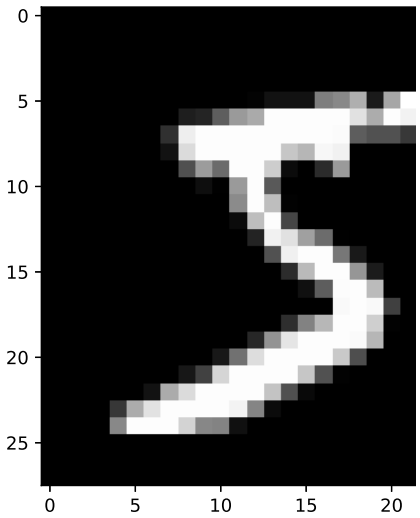


Figure: 数字の 5

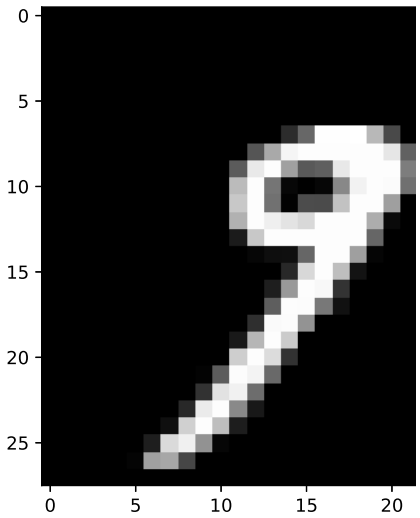


Figure: 数字の 9

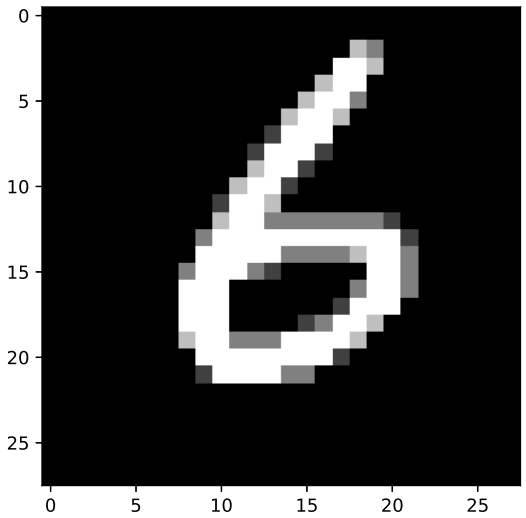


Figure: 数字の 6

学習結果

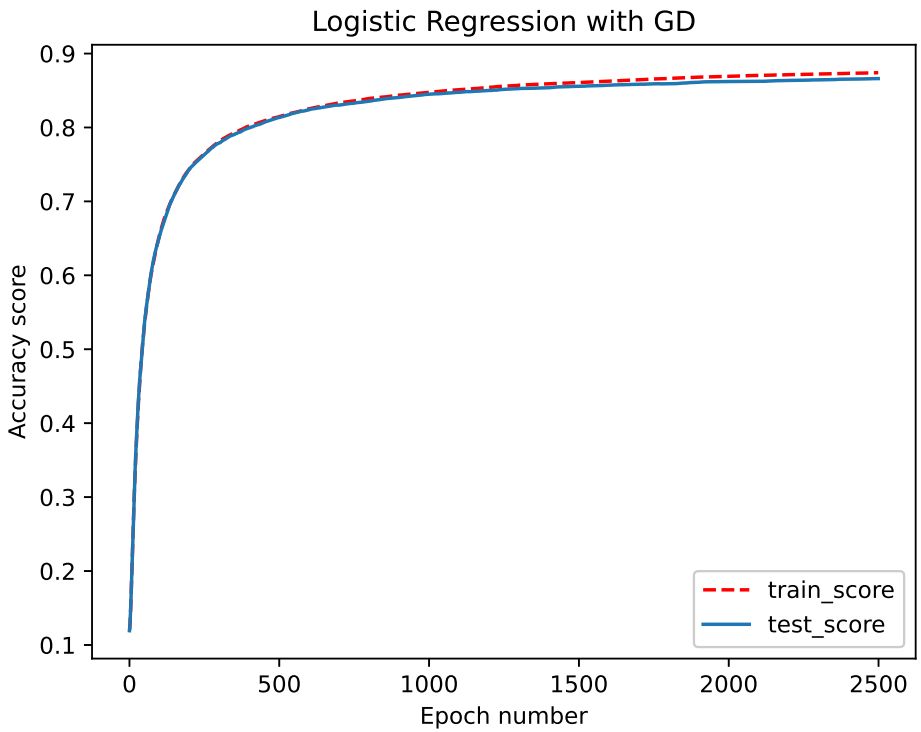


Figure: $\alpha = 0.0001$ とした時の, MNSIT 実験の結果

ロジスティック回帰で MNIST を学習した結果を左に示す.
テストデータに関しても 85% 程度の精度がでた.

Contents

活性化関数

深層学習について紹介する前に活性化関数について紹介する.

Definition (活性化関数)

$\sigma : \mathbb{R}^d \to \mathbb{R}^d$ を非線形かつ, *Lipschitz* 連続な関数とする. この時, σ を活性化関数 *(activation function)* という.

Example (活性化関数)

$$\sigma(x)_i = \frac{1}{1+\exp(-x_i)},$$
$$ReLU(x)_i = \max(x_i, 0).$$

σ をシグモイド関数 *(sigmoid function)*, *ReLU* を正規化線形関数 *(Rectified Linear Unit)* という.

回帰型ニューラルネットワーク

Definition (回帰型ニューラルネットワーク)

ML 空間 $(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{L}_D)$ を以下のように定義する.
$L = \{L_1, L_2, \cdots, L_K\}$ を自然数列とする. $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \mathbb{R}$,

$$\mathcal{H} = \left\{ f : \mathcal{X} \to \mathcal{Y} \;\middle|\; \begin{array}{l} f(x) = W^\top g_K \circ \cdots \circ g_1(x) + b, g_i = \eta(W_i x + b_i) \\ W_i \in \mathbb{R}^{L_i \times L_{i-1}}, b_i \in \mathbb{R}^{L_i}, W \in \mathbb{R}^{L_K}, b \in \mathbb{R} \end{array} \right\},$$

$$\mathcal{L}_D(f) = \frac{1}{N} \sum_{i=1}^{N} (f(x_i) - y_i)^2,$$

ここで η は活性化関数である. $f \in \mathcal{H}$ を回帰型 *(全結合)* ニューラルネットワーク *(Full-connected Neural Network for Regression)* と呼び, 各 g_i を f の第 i 層 *(layer)* と呼ぶ.

分類型ニューラルネットワーク

Definition (分類型ニューラルネットワーク)

ML 空間 $(\mathcal{X}, \mathcal{Y}, D, \mathcal{H}, \mathcal{L}_D)$ を以下のように定義する.
$L = \{L_1, L_2, \cdots, L_K\}$ を自然数列とする. $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \mathbb{R}^m$,

$$\mathcal{H} = \left\{ f : \mathcal{X} \to \mathcal{Y} \;\middle|\; \begin{array}{l} f(x) = \psi(W g_K \circ \cdots \circ g_1(x) + b), g_i = \eta(W_i x + b_i) \\ W_i \in \mathbb{R}^{L_i \times L_{i-1}}, b_i \in \mathbb{R}^{L_i}, W \in \mathbb{R}^{m \times L_K}, b \in \mathbb{R}^m \end{array} \right\},$$

$$\mathcal{L}_D(f) = -\sum_{n=1}^{N}\sum_{k=1}^{m} y_{nk} \log f(x_n)_k,$$

ここで η は活性化関数, y_i は *one-hot* ベクトル, ψ はソフトマックス関数である. $f \in \mathcal{H}$ を分類型 (全結合) ニューラルネットワーク *(Full-connected Neural Network for Classification)* と呼び, 回帰型の時と同様に各 g_i を f の第 i 層と呼ぶ.

深層学習

以下, ニューラルネットと書けば, 分類, 回帰ニューラルネットワークのいずれかを表すものとする.

Definition (Deep Learning)

ニューラルネットワークが $K \geq 3$ の時に

$$\arg\min_{f \in \mathcal{H}} \mathcal{L}_D(f)$$

を求める問題を深層学習 *(Deep Learning)* と呼ぶ.

Contents

Universal Approximation Theorem

ニューラルネットの近似能力に関する定理としては, 以下のような定理が知られている.

Theorem (普遍性定理 [1])

$\mathcal{X} = I_n$, $\mathcal{H}$ を $K = 1$ の時の回帰型ニューラルネットの仮説空間とする. この時, 活性化関数 η がシグモイド的関数ならば,

$$\forall f \in C(\mathcal{X}), \forall \varepsilon > 0, \exists G \in \mathcal{H} \ s.t. \ \sup_{x \in \mathcal{X}} |f(x) - G(x)| < \varepsilon$$

が成立する.

証明には Hahn・Banach の定理と Liesz の表現定理を用いる.

理論 vs. 実験

ニューラルネットワークについてはまだまだわかっていないことが多く, 理論と実験結果が合わないようなこともしばしばある. 具体的には, 以下のようなものがある.

1. 学習がうまくいく (深層学習の目的関数は非凸関数 (局所最適解やプラトーが多く存在する))
2. 層を増やしたほうが精度が向上する (一般に, モデルを複雑にしすぎると過学習を引き起こす)

References

[1] G. Cybenko, Approximation by Superpositions of a Sigmoidal Function, Mathmatics of control, signal and systems, vol. 2, no. 4, 1989

[2] Preferred Networks, ディープラーニング入門 Chainer チュートリアル, https://tutorials.chainer.org/ja/index.html, 2019